DOCUMENT RESUME

ED 134 158 IR 004 317

AUTHOR Dietsche, Peggy; Gehler, Ron
TITLE Structured Programming for the Unstructured Student; An Individualized Approach?
PUB DATE Oct 76
NOTE 18p.; Paper presented at National Association of Users of Computer Applications to Learning (Portland, Oregon, October, 1976)

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Computer Oriented Programs; Educational Methods; Individualized Instruction; Program Design; Program Development; *Programed Instruction; Programed Materials; *Programing
IDENTIFIERS COBOL

ABSTRACT

The individualized instructional model used is based on fixed-content, variable time, and fixed proficiency. Flowcharting is retained along with decision tables as part of the training. Top-down modular design is introduced when students are well advanced. Five tasks are identified for use in one individualized system. The first three involve knowing and doing competencies, and the other two concentrate on applying the knowledge in writing well-structured self-documenting programs. Learning packages including a learning guide, resources, and written criterion examination are designed to teach each task. With each packet the student receives a contract to be signed. When mastery of the program is obtained the student is able to write well-structured (top-down, modular) programs. (WBC)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

ED134158

STRUCTURED PROGRAMMING

FOR THE UNSTRUCTURED STUDENT

AN INDIVIDUALIZED APPROACH

T ? F

PRESENTATION BY

PEGGY DIETSCHE and RON GEHLER

PERMISSION TO REPRODUCE THIS COPYRIGHTED MATERIAL HAS BEEN GRANTED BY

Peggy Stumer Dietsche
Ron Gehler

TO ERIC AND ORGANIZATIONS OPERATING UNDER AGREEMENTS WITH THE NATIONAL INSTITUTE OF EDUCATION. FURTHER REPRODUCTION OUTSIDE THE ERIC SYSTEM REQUIRES PERMISSION OF THE COPYRIGHT OWNER.

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

IR004317

Introduction

Structured programming is not only the "in" thing to do today, but also a whole new programming methodology which is here to stay! In addition to being frequently discussed both verbally and in written form, it is coming into widespread use due to its proven superiority to other techniques. Structured programming has evolved in response to the continual increase in size and complexity of computer programming projects. But this evolution has resulted in building and maintaining costly systems.

This is where structured programming begins, with concise, clear and easily maintained programs. It imposes a strict discipline on the design of the overall program, using top-down, modular techniques.

Although many articles have been published regarding this subject, there have been too few textbooks dedicated to exploring the techniques of structured programming. Therefore, no one textbook could satisfy our requirements for a training package on structured programming that would be an individualized competency based instruction. We have developed this package more fully, using films, textbooks and exercises as well as other specialized tasks. Students who master all five tasks must demonstrate structure, modularity, Hierarchy plus Input Process Output (HIPO), structure charts/trees, pseudo code and structured walk throughs.

This series of tasks is unique because the time for transition from unstructured to structured techniques will vary for any given student. The approach used in this package gives the student flexibility in making this transition.

Objectives

The objectives for this paper are as follows:

(1) Identify the use of individualized instruction at the Metropolitan Vocational Education Consortium (MVEC) career Electronic Data Processing (EDP) training center.

(2) Identify structured programming training for students at MVEC career EDP training center.

(3) Communicate the method of teaching structured programming through an individualized approach.

Metropolitan Vocational Education Consortium (MVEC)

The Metropolitan Vocational Education Consortium (MVEC) is a joint powers agreement among seven of the eight vocational technical institutes in the Minneapolis, St. Paul and surrounding area. This agreement, established in May 1974, permits the new organization (MVEC) to cooperatively provide an effective delivery system of electronic data processing (EDP) instructional resources to vocational technical students. MVEC is a vocational instructional resource to the overall Minnesota Educational Computing Consortium.

916 Area Vocational Technical Institute (AVTI), located a few miles northeast of St. Paul, is the location of the EDP career instructional center for MVEC. This career center has enrolled 120 post high school and 135 adult extension students. The EDP staff at this center have developed the individualized structured programming package which is addressed in this paper.

Individualized Instruction

Individualized instruction is a widely used term in education today and means almost any type of instruction that responds to the unique needs of the individual.

The individualized instructional model used at 916 AVTI is based on fixed-content, variable time and fixed proficiency. This model seems to work well in vocational education where the main objective is to assist students in the development of skills and knowledge to enter an occupation regardless of the amount of time it takes an individual. Individualized instruction will be covered in more depth later in this paper as we examine the parts and content of the individualized package on structured programming.

Problem-Solving Tools Used in Our Programming Training

Detail flowcharting using the American National Standards (ANS) symbols or symbols similar to ANS has been the tool used by programmers for over fifteen years. This tool has had varying degrees of success as we moved from first generation computers through third generation computers. Some programmers use detail flowcharts before they code; others feel flowcharting is no longer needed since most programming today is done with the high level macro languages such as COBOL, FORTRAN and PL1.

Detail flowcharting using ANS symbols is still part of our programmer training at 916 AVTI. We see this tool receiving less emphasis in our training as we evolve toward structured programming tools and techniques.

Decision Tables

The use of decision tables has become a very useful problem-solving tool to both programmers and systems analysts. Decision tables can usually be used instead of flowcharts in program design. The use of decision tables has the following advantages over flowcharts:

(1) Difficult problems can be handled more easily.

(2) Tables force program modularity.

(3) Tables are generally easier to create, to change and to read.

(4) Condition-testing is separated from the action-taking, thus making it easier to follow an exact path to completion.

We continue to use decision tables in our training of programmers. Decision tables will probably continue to have strong emphasis as we evolve toward structured programming training for students.

Top-Down Modular Design

One of the many techniques used in structured programming is top-down design. This technique has been used in many EDP shops in the past few years and has often been called top-down modular design or modular dependency charts.

Top-down design has been part of our programmer training but was not introduced to the students until they were well advanced in their programing training. We now plan to introduce this technique to our students at the same time as they learn other problem-solving tools such as flowcharting and decision tables.

Structured Programming Training Background Problems

Our instructional advisory committee represents both large and small EDP shops as well as many varied industries, both public and private. This committee recommended that we add structured programming to our training, but that we still retain modular programming, decision tables and standard flowcharting techniques. They agreed that, while many EDP shops are trying structured programming and implementing these techniques, most shops still have not gone this route.

Until 1976, obtaining instructional materials for training students in structured programming was a problem. There have been a large number of papers written on this subject in recent years, but very few books have explored the topic.

Another problem we faced was instructor training. None of our instructors had used structured programming during their years in industry. Most of our programming instructors have used or were familiar with modular programming techniques, so we feel that they will adapt to structured programming techniques as they work with the individualized instructional packets that we have created on structured programming.

Overview: Five Structured Programming Packets

There are five tasks identified for use in one individualized system for teaching structured programming. The first three tasks involve *knowing* and *doing* competencies in structured programming concepts, techniques and theory. These first three tasks, when mastered, will take sixty hours for the average student to complete. The remaining two tasks concentrate on *applying* what was learned in the first three tasks by writing well structured, self-documenting programs for a total of eighty-five hours. The five tasks are:

(1) Demonstrate introductory structured programming concepts I.

(2) Demonstrate structured programming techniques in COBOL II.

(3) Use advanced structured programming techniques in COBOL II.

(4) Solve a business problem using structured programming in COBOL.

(5) Solve business problems using structured programming in COBOL.

Learning Guide Format

Once the tasks are identified, learning packages are designed to teach each of these job tasks. Each learning package is composed of the following:

(1) A Learning Guide to give the students directions on how to learn each task.

(2) Resources such as information and procedure sheets, audio-visual presentations and textbooks. These resources actually teach the student how to do the specific job task for which the package is designed.

(3) Written Criterion Examinations (if required) to determine whether the student can perform the task to a preset standard.

The learning packages, then, contain the meat of what the student must learn and the directions for how to learn the job tasks.

Front Cover of the Learning Guide

The front cover of a learning guide tells the student:

(1) the task name.

(2) the task number.

(3) the purpose for mastering the task.

The task statement which begins with a measurable action verb explains to the student what the learning package will teach her/him to do. This statement appears on the cover of the learning package along with a unique number for its identification.

The purpose relates the task to the occupation; it justifies the student's time in mastering the task. This explanation provides the rationale to motivate the student to learn the task. Each packet is identified at the bottom with a department number, program number, task number and a pre-requisite task number.

Contract Page

The student receives a contract to read and sign for each packet. This contract is an agreement between the student and the program instructor to complete the task. In addition to this agreement, the contract:

(1) states the total number of hours it will take an average student to complete the task.

(2) defines a terminal performance objective.

(3) lists the micro-performance objectives.

The terminal performance objective, or TPO, contains the terms of the contract. The TPO defines for the student:

(1) the conditions of the performance (what the student will be able to use to do the task for evaluation).

(2) the desired outcome or behavior.

(3) the means used to evaluate mastery.

(4) the level or standard of performance.

In short, the TPO tells the student exactly what she/he must do to perform this task successfully.

The micro-performance objectives, or MPOs, divide the TPO into workable, logical units which, if completed correctly, accomplish the task objective. Each MPO is numbered consecutively. These MPOs, when combined, accomplish the TPO. The MPOs tell the student the steps she/he must complete in order to master the TPO, and clearly organizes the information in the packet for the student and instructor.

The MPO Page

For each micro-performance objective listed on the contract, there is an MPO page in the packet. The purpose of the MPO page is to direct the student through the learning activities. This page restates the MPO and lists several learning steps and resources that must be followed in a specified order if the student is to complete the particular MPO directions.

The learning steps are point-by-point instructions used to accomplish the MPO. These learning steps are directives which tell the student what to do and offer a rationale for the activity. All the learning steps equal the MPO just as all MPOs equal the TPO. Every learning step refers to a resource located directly across from the learning step and is identified with a corresponding number. A variety of resources is used whenever possible because more than one resource per learning step accomodates diverse learning styles.

The resource lists the materials, ie., audio-visual, textbook, needed for the student to complete the learning steps. These resources are written for easy identification by the student and tell the student where to acquire the materials.

For further illustrations of the learning guide, front cover, contract page, and MPO page, consult the appendix.

Five Structured Programming Packets

The student should have substantial programming experience to accomplish these tasks since they do not teach a language but, instead, teach techniques and program structure. It is also beneficial to the student if she/he is able to write top-down modular COBOL programs. However, PL/1 or FORTRAN could be substituted. The intent of these packets, then, is to concentrate on structured tools, techniques, control structure, and knowing and applying these methods to the programs.

First Structured Programming Packet

TASK: #275 Demonstrate introductory structured programming Concepts I.

PURPOSE: Write concise, clear and easily maintained programs through the use of restricted control logic. This style of programming is known as structured programming.

TIME: 15 Hours

TPO: Given the texts and audio-visual materials, demonstrate introductory structured programming concepts to 100% accuracy on the criterion examination.

MPO 1: Demonstrate basic principles and practices used in creating well-structured programs.

MPO 2: Demonstrate good programming practices using the principle of consistency and reconstruct code.

MPO 3: Define the formalized programming style of structured programming.

MPO 4: Define the history, objectives, techniques and theory of structured programming according to Edward Yourdon.

This task introduces the student to structured programming concepts at an elementary level. Two video-tapes, "Critical Program Reading", Parts 1 & 2 by Edutronics, produced under the direction of Gerald M. Weinberg, are used along with the corresponding Edutronics workbook. Portions of the text, Installation Management, by IBM are read, along with chapters from Techniques of Program Structure and Design by Edward Yourdon.

The student completes various exercises, ie., self-tests, reconstructing code and eleminate the "Go to's", and constructs a structured programming notebook to define Hierarchy plus Input Process Output (HIPO), control logic structures, structured walk-throughs, chief programmer team concept, and psuedo code. History, background, theory and techniques of structured programming are also studied.

Three multiple choice, open book criterion examinations, one after MPO 1 and MPO 2, another after MPO 3 and another after MPO 4, must be completed to 100% accuracy before the student can master this task; two retakes are allowed. When mastered, the student may go on to Task #276.

Second Structured Programming Packet

TASK: #276 Demonstrate structured programming techniques in COBOL II.

PURPOSE: Write concise, clear and easily maintained programs through the use of restricted control logic. This style of programming is known as structured programming.

TIME: 20 Hours

TPO: Given texts and audio-visual materials, demonstrate top-down design in structured programming to 80% accuracy on the criterion examination and 100% accuracy on the exercises.

MPO 1: Demonstrate the principles and tools of top-down design in structured programming.

MPO 2: Develop improved design techniques using the principles and tools of top-down design in structured programming.

MPO 3: Discuss a variety of opinions and approaches to structured programming.

This task builds on information learned in the previous task. It also applies structured techniques to the COBOL language. Top-down design and modularity are emphasized, and tools such as structure charts, structure trees and psuedo code are dealt with in more detail.

The student views two Edutronics video-tapes, "Top-Down Design", Parts 1 & 2, and reads the Edutronics workbook. Readings from Reliable Software Through Composite Design by Glenford Myers and two current articles concerning structured programming are included for student instruction. The student also completes exercises which include: writing an algorithm psuedo code, designing a structure chart and structure tree.

Readings from another text and two more current articles on structured programming, of the student's choice, are summarized and included in their structured programming notebook begun in the previous packet. This notebook is used throughout this series of tasks by the student for taking notes and future references.

One closed book criterion examination must be completed to 80% accuracy and the student's notebook must receive a 100% accuracy rating by the instructor before mastery of this task is obtained. The student then goes on to Task #277.

Third Structured Programming Packet

TASK: #277 Use advanced structured programming techniques in COBOL II.

PURPOSE: In order to make impressive gains in program reliability and maintainability, a programmer must recognize that structured programming imposes a strict discipline on the design of the overall program.

TIME: 25 Hours

TPO: Given the texts and audio-visual materials, demonstrate advanced control structures and structured programming techniques to 100% accuracy on the exercises.

MPO 1: Apply "careful pre-coding planning" for total programming efficiency.

MPO 2: Recognize properly constructed conditions, structures and designs.

MPO 3: Identify HIPO as part of the improved programming technologies.

MPO 4: Identify well-structured design.

MPO 5: Demonstrate a structured walk-through using a structure chart.

Notice that the purpose in this task has changed somewhat from the previous two tasks. Control structure, design and structured techniques are studied at a more advanced level.

Two video-tapes are viewed, "Control Structure", Parts 1 & 2, by Edutronics. Readings include: Edutronics workbooks, Installation Management by IBM, Structured Programming Student Handbook by Burroughs Corporation, current articles, and a Navy manual, Structured Programming Using COBOL. Activities to complete include: coding a nested "if" statement, illustrating a HIPO diagram, summarizing concepts of proof of correctness, structured flowcharts, structured walk-throughs, coupling, binding and levels of abstraction.

A typical problem which is given to the student involves a nested "if" statement. The student is requested to construct a structure chart and demonstrate a structured walk-through to an instructor. The student, upon completion and mastery of this task, should be making productive gains using control structure and top-down design and should be ready to apply this know-how in the next packet.

Fourth Structured Programming Packet

TASK: #278 Solve a business problem using structured programming in COBOL.

PURPOSE: In order to make productive gains in program reliability and maintainability, structured programming must impose a strict discipline on the design of the overall program.

TIME: 25 Hours

TPO: Given a business application problem and the computer, the student will design and create input layouts, output layouts, HIPO diagram, structure chart, structure tree and psuedo code; she/he will code a structured COBOL, self-documenting program, compile and test the program on the computer. 100% accuracy required on the results of the problem.

MPO 1: Define self-documenting structured COBOL programs.

MPO 2: Solve a business problem using structured COBOL programming.

The first MPO in this task deals with identifying techniques to enhance the readability of a program, e.g., descriptive data names, indentation, program narration. The student needs to determine program relationships, functions of indentation, use of sections, and program narration.

Installation Management by IBM is read and the Edutronics video-tapes, "Critical Program Reading", Parts I & II, are viewed by the student, and sample self-documenting COBOL programs are available.

The second MPO involves a business application problem which the student must solve. It is a problem of medium difficulty which involves a sort and generates a report. The student completes the following activities as stated in the TPO: input layout, output layout, HIPO diagram, general flowchart (top-down, modular), structure chart and structure tree, psuedo code. She/he also codes and compiles a self-documenting program, creates test data that will test every module and all possible conditions.

Normally the sutdent is not required to use all the structured tools listed above, they are used here to give the student practice and to give her/him a comparison of various methods of programming.

Upon completion of the program and activities to 100% accuracy, the student may go on to the last packet in this series of structured programming tasks.

Fifth Structured Programming Packet

TASK: #279 Solve business problems using structured programming in COBOL.

PURPOSE: In order to make productive gains in program reliability and maintainability, structured programming must impose a strict discipline on the design of the overall program.

TIME: 60 Hours

TPO: Given three business application problems and the computer, design and create input layouts, output layouts, HIPO diagrams, structure charts, structure trees, psuedo code. Code structured COBOL self-documenting programs, compile and test the programs on the computer. Obtain 100% accuracy on the results of each program.

MPO 1: Solve a hospital problem using the structured programming techniques.

MPO 2: Solve a checking application problem using the structured technique.

MPO 3: Solve an inventory update problem using the structured technique.

When mastery on this task is obtained, the student is able to write well-structured (top-down, modular) programs that are concise, clear and 100% reliable. This series of structured programming tasks prepares the student, at entry level, to use structured programming techniques in a real-world situation.

Conclusion

An individualized approach to structured programming? Yes! It does indeed work and can be considered a personalized, self-paced instruction. This heuristic series of five structured programming packets guides the student to accomlish an objective through the use of various media, exercises and tests. At the same time, allowing the student freedom to learn independently from resources of her/his own choosing. The packets are flexible, enabling both the fast learner and the slow learner to obtain the same skill upon completion.

VO-TECH INSTITUTE
3300 Century Avenue North
White Bear Lake, Minnesota 55110

A LEARNING GUIDE

TASK # 277 USE ADVANCED STRUCTURED PROGRAMMING TECHNIQUES IN COBOL III

PURPOSE TO MAKE IMPRESSIVE GAINS IN PROGRAM RELIABILITY AND MAINTAINABILITY, STRUCTURED PROGRAMMING IMPOSES A STRICT DISCIPLINE ON THE DESIGN OF THE OVERALL PROGRAM.

Reviewed by [illegible]

Validated by [illegible] 5-18-76

Dept.	Prog.	Task	Prereq.
500	502	277	276

© 916 Vo-Tech 1975

LEARNING CONTRACT

1 STUDENT DATA

NAME _____ SOCIAL SECURITY NUMBER _____

LENGTH OF CONTRACT (NORMAL TIME IN HOURS) 25 hours

2 TERMINAL PERFORMANCE OBJECTIVE

Given the texts and A-V materials, demonstrate advanced control structures and structured programming techniques to 100% accuracy on the exercises.

2a Micro-Performance Objective(s)

1. Apply careful pre-coding planning for total programming efficiency.
2. Recognize properly constructed conditionals, structure & design.
3. Identify HIPO as part of the Improved Programming Technologies.
4. Identify good structured design.
5. Demonstrate a structured walk-through using a structure chart.

3 AGREEMENT

I, _____ agree to complete the above stated terminal performance requirement within _____ to _____. I further recognize that the conditions of the contract (performance and time agreement) report my ability to perform the requirements of the occupation and record my progress.

Student's Signature

Dept.	Prog.	Task	Prereq
500	502	277	276

Instructor's Signature (verifies competency)

© 916 Vo-Tech 1972

MICRO - PERFORMANCE OBJECTIVE MPO 1

Apply careful pre-coding planning for total programming efficiency.

LEARNING STEPS

1. Read Resource 1a. and view Resource 1b. for an introduction to control structures.
2. Read Resource 3 for a discussion of coding changes & techniques.
3. Read Resource 3 to identify structure and control and develop pre-coding planning.
4. Study Resource 4a. to complete the exercise in Resource 4b.
5. Ask for Resource 5a. to be signed where applicable by an instructor.
6. Hand in the above exercises to be corrected; then include them in Resource 6.
7. Go on to MPO 2.

RESOURCES

1a. Pg. 9 Edutronic Workbook #434 & #435, Control Structures, found in LRC.

1b. Edutronics film #434 & #435, Control Structures I, found in LRC.

2. Pp. 13-18, Edutronic Workbook #434 & #435, Control Structures.

3. Pp. 19-35, Edutronic Workbook #434 & #435, Control Structures.

4a. Pp. 37-39, Edutronic Workbook #434 & #435, Control Structures.

4b. MVEC style sheet and exercise found in business data processing classroom.

5a. Structured Technique's Checklist, Task 277 found in business data processing classroom.

5b. Cobol coding form.

6. Your Structured Programming Review notebook.

Dept.	Prog.	Task	TPO	MPO
500	502	277	277	001

BIBLIOGRAPHY

TEXTS:

Baird, George, Department of the Navy, Structured Programming Using COBOL, (ANSI COBOL) Fadpug Presentation, ADPE Section Office, Washington D.C. 20376, 13 March 1974.

Burroughs Corporation, Structured Programming Student Handbook, Detroit, Michigan 48232, 1975.

IBM, Installation Management, An Introduction to Structured Programming in COBOL, GC20-1776-0, 1133 Westchester Avenue, White Plains, New York 10604, 1776.

Kleid, Naomi G., Critical Program Reading, Parts I & II, workbook, Lessons #430 and #431, Edutronics System International, Inc., 3435 Broadway, Kansas City, Missouri 64111, 1976.

Kleid, Naomi G., Top-Down Design, Parts I & II, workbook, Lessons #432 and #433, Edutronics System International, Inc., 3435 Broadway, Kansas City, Missouri 64111, 1976.

Kleid, Naomi G., Control Structures, Parts I and II, workbook, Lessons #434 and #435, Edutronics System International, Inc., 3435 Broadway, Kansas City, Missouri 64111, 1976.

Myers, Glenford J., Reliable Software Through Composite Design, Petrocelli, Charter, New York, 1975.

Pucel, Daird J. and Knaak, William C., Individualized Vocational and Technical Instruction, Charles E. Merrill Publishing Company, 1300 Alum Creek Drive, Columbus, Ohio 43216, 1975.

Yourdon, Edward, Techniques of Program Structure and Design, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1975.

AUDIO-VISUAL MATERIALS:

Edutronics, "Critical Program Reading" Part I, #430, 1976.

Edutronics, "Critical Program Reading" Part II, #431, 1976.

Edutronics, "Top-Down Design" Part I, #432, 1976.

Edutronics, "Top-Down Design" Part II, #433, 1976.

Edutronics, "Control Structures" Part I, #434, 1976.

Edutronics, "Control Structures" Part II, #435, 1976.

ARTICLES:

Babcock, Perry, "Outline Form of Structure Rules Provides Essentials", Computerworld, 23 February 1976.

Inmon, Bill, "An Example of Structured Design", Datamation, March 1976.

Jones, Martha Nyball, "HIPO for Developing Specifications", Datamation, March 1976.

Nilges, Edward G., "Structured Code Can be Built - Even With COBOL", Compuceworld, 22 March 1976.